PhySIC: A Veto Supertree Method with Desirable Properties

VINCENT RANWEZ, VINCENT BERRY, ALEXIS CRISCUOLO, PIERRE-HENRI FABRE, SYLVAIN GUILLEMOT, CELINE SCORNAVACCA, AND EMMANUEL J. P. DOUZERY

1 Institut des Sciences de l’Evolution (ISEM, UMR 5554 CNRS), Université Montpellier II, Place E. Bataillon, CC 064, 34095 Montpellier Cedex 5, France; E-mail: ranwez@isem.univ-montp2.fr (V.R.)

2 Laboratoire d’Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM, UMR 5506, CNRS), Université Montpellier II 161, rue Ada, 34392 Montpellier Cedex 5, France

Abstract.—This paper focuses on veto supertree methods; i.e., methods that aim at producing a conservative synthesis of the relationships agreed upon by all source trees. We propose desirable properties that a supertree should satisfy in this framework, namely the non-contradiction property (PC) and the induction property (PI). The former requires that the supertree does not contain relationships that contradict one or a combination of the source topologies, whereas the latter requires that all topological information contained in the supertree is present in a source tree or collectively induced by several source trees. We provide simple examples to illustrate their relevance and that allow a comparison with previously advocated properties. We show that these properties can be checked in polynomial time for any given rooted supertree. Moreover, we introduce the PhySIC method (PHYlogenetic Signal with induction and non-Contradiction). For k input trees spanning a set of n taxa, this method produces a supertree that satisfies the above-mentioned properties in O(kn^3 + n!) computing time. The polytomies of the produced supertree are also tagged by labels indicating areas of conflict as well as those with insufficient overlap. As a whole, PhySIC enables the user to quickly summarize consensual information of a set of trees and localize groups of taxa for which the data require consolidation. Lastly, we illustrate the behaviour of PhySIC on primate data sets of various sizes, and propose a supertree covering 95% of all primate extant genera. The PhySIC algorithm is available at http://atgc.lirmm.fr/cgi-bin/PhySIC. [Formal properties; phylogenetics; polynomial-time algorithms; primates; software; supertrees; triplets; veto methods.]

Building Supertrees

Phylogenies are invaluable tools in various areas of biology to understand the evolution of genes and taxa. Trees that incorporate an exhaustive sampling of taxonomic biodiversity provide crucial information about systematics, genomics, and diversification patterns of species (e.g., Davies et al., 2004). Large trees can be built using various approaches, including supertrees and supertrees. The former approach consists of combining the different source data sets into a supermatrix of characters and then analyzing it under standard phylogenetic reconstruction criteria (e.g., Delsuc et al., 2005). The supertree approach is an alternative methodology using trees rather than character data as a primary source of information. It first involves inferring partially overlapping, source phylogenetic trees from initial character data and then assembling them into a larger, more comprehensive supertree (Bininda-Emonds, 2004a). This approach is particularly convenient when dealing with heterogeneous character sources; e.g., those scored from morphological, transposable elements, DNA, or protein studies. Supertrees have become increasingly popular (e.g., Bininda-Emonds, 2004b), notably since the seminal work involving the reconstruction of the primate supertree (Purvis, 1995a). The widespread use of supertrees is explained by three useful applications (Wilkinson et al., 2004): (i) they provide large phylogenetic frameworks for broad comparative studies; (ii) they evaluate the congruence of sets of input trees, and reveal conflicts due to outlier/unstable taxa; and (iii) they identify insufficient overlap among leaf sets of input trees and assign priorities for choosing the taxa to be subsequently sampled.

Different Kinds of Supertree Methods

Supertree methods fall into three categories depending on their way of handling topological conflicts; i.e., different arrangements of the same leaves among labeled source trees.

The first suite of methods does not handle incompatible source trees. The pioneering methods that belong to this category are Build (Aho et al., 1981) and the strict consensus supertree (Gordon, 1986). Although they are important milestones, these methods appear "of limited use. As most systematists know, phylogenies usually conflict with one another" (Bininda-Emonds, 2004b:4).

The second suite of methods handles conflicts among input trees in a liberal way: they apply a voting procedure. In order to extract their main phylogenetic signal, source trees are asked to vote on various parts of the phylogeny to be inferred, with the most-supported candidates being elected and composing the output supertree. Voting methods are said to resolve conflicts (Thorley and Wilkinson, 2003): for each conflict, they use some optimization criterion to make a decision in favor of one of the topological alternatives. Most conflicts among input trees are expected to be resolved because relationships displayed by the supertree are guided by source topologies on the basis of weight of evidence. The most widespread voting method is matrix representation with parsimony (MRP) whereby nodes of each source tree are encoded as binary characters of a matrix, which is then analyzed with the maximum
parsimony criterion to obtain the composite tree (Baum, 1992; Ragan, 1992). Analyzing this binary encoding of source topological information with other tree-building criteria leads to variants of MRP such as matrix representation with flipping (MRF; Chen et al., 2003) and matrix representation with compatibility (MRC; Ross and Rodrigo, 2004). Other methods of the voting kind, such as MinCut (MC; Semple and Steel, 2000) and Modified-MinCut (MMC; Page, 2002) extend Build. They encode source trees in a graph that is progressively decomposed to get supertree clades. When conflicts hinder the decomposition, the graph is cut by removing the least supported relationships. The Average Consensus Supertree (Lapointe and Cucunel, 1997) and super distance matrix (Criscuolo et al., 2006) methods implement the voting approach in an alternative way. They average the initial distance matrices, converted from source characters or valued topologies, into a superdistance matrix; a tree-building distance-based approach is then used to infer a supertree from this matrix. Interestingly, voting methods like MRP may generate novel clades; i.e., clades not present in any input tree alone (Purvis, 1995b; Bininda-Emonds and Bryant, 1998; Sanderson et al., 1998). Unfortunately, when source trees conflict, novel clades that are contradictory of each of the source trees can be present in the supertree inferred by MRP (Goloboff and Pol, 2002; Goloboff, 2005; Cotton et al., 2006) and by MRF (Goloboff, 2005). The importance of this phenomenon is still debated, Bininda-Emonds (2003) reporting, on the basis of simulations, that this situation is not very frequent for MRP, whereas Goloboff (2005) shows selected case studies where “this situation is, clearly, not very unlikely.”

The third suite of methods handles conflicts among input trees in a conservative way. They adopt a veto philosophy: the phylogenetic information of every source topology is to be respected, and the supertree is not allowed to contain clades that a source tree would vote against. These methods remove conflicts (Thorley and Wilkinson, 2003) because they either propose multifurcations in the supertree (Goloboff and Pol, 2002) or prune rogue taxa (Berry and Nicolas, 2004). In this framework, the supertree should retain a single branching pattern within a given clade when several valid topological alternatives are present in the source trees. The full agreement required by veto methods provides an unambiguous phylogenetic framework that is, for instance, well suited for taxonomic revisions. More specifically, such a conservative approach may be applied to automatically build or update parts of the Tree of Life (http://tolweb.org). Several supertree methods akin to the veto philosophy have been proposed, all of which are inspired by consensus approaches that operate on trees with identical leaf sets. For example, extensions of the strict consensus (Gordon, 1986; Huson et al., 1999), semi-strict consensus (Goloboff and Pol, 2002), and maximum agreement subtree consensus (Berry and Nicolas, 2004) have been proposed to infer veto supertrees.

Properties of Supertree Methods

To assess the relevance of supertree methods, it is most useful to have properties characterizing the extent to which the supertrees they infer are reliable syntheses of source trees (Bininda-Emonds and Bryant, 1998; Steel et al., 2000; Wilkinson et al., 2004; Goloboff, 2005). For instance, Steel et al. (2000) suggest that the output supertree should (i) encompass every source tree when possible, (ii) always contain every leaf (taxon) that occurs in at least one source tree, and (iii) be computed within a running time that grows polynomially with respect to the total number of leaves. These authors also showed that rooted input trees are more appealing than unrooted ones for supertree methods that aim to satisfy several desirable properties simultaneously. Yet, even if supertree methods satisfy some desirable properties, the inferred supertrees often contain polytomies that actually intermix two distinct phenomena: either a lack of overlap in the topological information among source trees, or the occurrence of topological conflicts among them, or a combination of these. We thus decided to develop a method that proposes supertrees with unambiguous resolutions, and provides biologists with explanations about causes of polytomies. For this purpose, we rely on two new formal properties.

On the one hand, we think that supertree methods should avoid arbitrary resolutions; i.e., resolutions that are not entailed by the source topologies. Indeed, novel relationships displayed by a supertree “are worrying if they are not implied by combinations of the input trees” (Wilkinson et al., 2005), and “should be identified as such, to highlight their lack of any known justification” (Pisani and Wilkinson, 2002). Thus, we first request that every piece of phylogenetic information displayed in the supertree be present in one or several source topologies or be induced by their interaction; we call this the induction property.

On the other hand, we focus on unanimous clades, thus adopting a veto point of view. This means that the supertree is not allowed to contain a clade that conflicts either directly with a source tree or indirectly with a combination of them. We call this the non-contradiction property. Such a supertree, which incorporates only uncontradicted input relationships, provides a reliable baseline for subsequent analyses (Goloboff and Pol, 2002; Goloboff, 2005).

Goloboff and Pol (2002) mentioned similar properties in a formal characterization involving triplets. They provide examples showing that supertree methods of the voting kind, such as MRP and MC, understandably do not respect these properties. Although being appealing, the characterization proposed by Goloboff and Pol (2002) can at times be too restrictive or permissive (see following sections). Recently, Grunewald et al. (2006) provided another characterization of a property related to arbitrary resolutions contained in the supertree with respect to source trees. In both cases, there does not seem to be any straightforward algorithm that would always allow
for property verification. Note, however, that Goloboff and Pol (2002) proposed a supertree heuristic algorithm that satisfies the desired properties in most cases.

In this paper, we provide a characterization of non-contradiction and induction properties that differs from those of Goloboff and Pol (2002) and Grunewald et al. (2006). We also describe simple and polynomial-time algorithms that enable users to check whether or not a given supertree satisfies these properties. Then we propose an algorithm called PhySIC that improves the Build algorithm (Aho et al., 1981) by always inferring a supertree and which, moreover, satisfies the non-contradiction and induction properties. As far as we know, this is the first time that a polynomial-time method is proposed that always satisfies properties related to induction and non-contradiction. Moreover, improving the behavior of Build with respect to arbitrary decisions can benefit the various methods that extend this algorithm for supertree purposes; e.g., MC (Semple and Steel, 2000), MCC (Page, 2002), AncestralBuild (Daniel and Semple, 2004; Berry and Semple, 2006), and RankedTree (Bryant et al., 2004). Next, we pinpoint the difference between the behavior of PhySIC and that of well-known supertree methods on a biological case study on Primates. Lastly, we illustrate PhySIC on the reconstruction of the primate supertree at the genus level from various source trees based on mitochondrial DNA, nuclear DNA, and jumping gene sequences. The supertree reconstructed appears to be useful for displaying phylogenetic relationships among the major primate taxa (Goodman et al., 1999). Moreover, the produced supertree displays label(s) on each of its polytomous nodes that identify the cause(s) of these polytomies (lack of cross-information and/or presence of contradictions). The PhySIC method has been implemented in C++ using the Bio++ library (Dutheil et al., 2006) and is freely available as a web service and for download at http://atgc.lirmm.fr/cgibin/PhySIC/physic.cgi.

Non-Contradiction and Induction Properties

We first introduce vocabulary and notations required to formally define the properties of non-contradiction (PC) and of induction (PI). Simple examples are then used to illustrate the relevance of PC and PI as well as to relate them with previously proposed properties for supertree methods (Steel et al., 2000; Goloboff and Pol, 2002). Then we show how to check in polynomial time whether a supertree satisfies PC and PI for a given collection of source trees.

Topological Description of Trees

The definitions and notations used for trees and their topological description are mainly the same as those used by Semple and Steel (2003). We only consider rooted phylogenies, due to the fact that supertree methods cannot fulfill different desirable properties listed in Steel et al. (2000) when considering unrooted trees. Hereafter, the terms phylogeny and tree are considered synonymous. Given a tree $T$, $L(T)$ denotes the set of taxa associated to its leaves. More generally, given a collection $T$ of trees, $L(T)$ denotes the set of taxa appearing in at least one tree of $T$. Given two phylogenies $T$ and $T'$ on the same leaf set ($L(T) = L(T')$), we say that $T$ refines $T'$ whenever $T$ contains all clades of $T'$. In other words, either $T$ and $T'$ are identical or $T$ can be transformed into $T'$ by collapsing some of its internal edges.

A rooted tree on three leaves $A$, $B$, $C$ has only three possible binary shapes, called triplets and denoted by $ABC$, respectively $ACB$, respectively $BCA$, depending on the innermost clade ($AB$, respectively $AC$, respectively $BC$). Given a triplet $t$, $I(t)$ denotes any of the two other triplets on the same set of leaves. Alternatively, a tree on three leaves can be a star tree; i.e., a unique internal node connected to the leaves. Any rooted tree $T$ can be equivalently described by the set of triplets homeomorphic to subtrees of $T$ connecting three leaves (e.g., Grunewald et al., 2006); $rt(T)$ denotes this set. Given a collection $T$ of phylogenies, $rt(T) = \bigcup_{t \in T} rt(t)$ denotes the set of triplets present in these phylogenies. Note that it is possible that $rt(T)$ contains two triplets $t$ and $I(t)$, namely when $T$ hosts two incompatible phylogenies. Clearly, two such triplets cannot be combined into a single supertree of the collection.

Given a set $\mathcal{R}$ of triplets, $L(\mathcal{R})$ denotes the set of taxa appearing in at least one tree in $\mathcal{R}$. A tree $T$ is said to display a set $\mathcal{R}$ of triplets when $\mathcal{R} \subseteq rt(T)$; moreover, $T$ strictly displays $\mathcal{R}$ if additionally $L(T) = L(\mathcal{R})$. A set $\mathcal{R}$ of triplets is compatible if there is a tree $T$ that displays $\mathcal{R}$. To find a tree displaying $\mathcal{R}$, it is useful to take into account that some triplets of the tree are induced by $\mathcal{R}$: a compatible set $\mathcal{R}$ of triplets induces a triplet $t$, denoted by $\mathcal{R} \vdash t$, and only if $\mathcal{R} \cup \{t\}$ is not compatible, or equivalently if any tree $T$ that displays $\mathcal{R}$ contains $t$. For instance, any tree displaying $\{AB|C, BC|D\}$ also has to display the triplet $AC|D$; i.e., $\{AB|C, BC|D\} \Rightarrow AC|D$. Bandelt and Dress (1986) and Dekker (1986) were among the first to investigate such induction rules. The set of all triplets induced by a compatible set $\mathcal{R}$ is called the closure of $\mathcal{R}$ and is denoted by $cl(\mathcal{R})$. Source trees considered for supertree building are sometimes incompatible, and then the set of triplets considered is incompatible. Nonetheless, we can characterize the set of triplets induced by these collections by extending the preceding definition: we will say that a set $\mathcal{R}$ of triplets induces a triplet $t$ when there is a compatible subset $\mathcal{R}'$ of $\mathcal{R}$ that induces $t$.

Characterizing Non-Contradiction and Induction by Triplets

Here we describe two important properties that veto method supertrees should satisfy. They concern topological relationships that a supertree should not contain with respect to the input trees: first, it should not contain relationships contradicting the source trees (PC property); moreover, it should only contain relationships that are induced by the input trees (PI property). Below we detail these two properties.

There are several ways for a supertree to contradict a collection of source trees. The most direct contradiction...
occurs when only one resolution appears for a group of taxa in one or several source trees, and the supertree contains a different resolution for the group. When different resolutions appear in source trees, as soon as the supertree proposes a resolution for the concerned taxa it contradicts at least one input tree. Contradictions are less direct when the supertree proposes a resolution that contradicts no single input tree but does contradict a combination of them.

A relationship contained in a supertree can present no contradiction with the input trees and still not be desirable. For instance, Figure 1 shows a collection of two source trees and four possible supertrees, $T'$, $T''$, $T'''$, and $T$. Unlike $T$, the supertrees $T'$, $T''$, $T'''$ propose an arbitrary resolution for the clade, $A$, $B$, $C$.

For instance, considering collection $T = \{T_1, T_2\}$ in Figure 2 and supertree $T'$ of Figure 3, the set $\mathcal{R}(T', T)$ is \{AC|E, AC|F, AB|E, AB|F, BC|E, BC|F, EF|A, EF|B, EF|C\}. Note that the triplet $AD|C$ present in $rt(T')$ due to $T_2$ is not in this set because $A$, $D$, $C$ are multifurcating in $T'$. When the source trees are incompatible, it is possible that $\mathcal{R}(T, T)$ contains two different triplets for the same three taxa. For example, consider the supertree $T$ in Figure 3 proposed by the MC and MMC voting methods (Semple and Steel, 2000; Page, 2002) on the collection $T = \{T_1, T_2\}$. $\mathcal{R}(T, T)$ contains both $AB|C$ (resulting from $T_2$) and $AC|B$ (resulting from $T_1$). In this case, the supertree $T$ that contains the triplet $t = AB|C$ does not satisfy PC, since $t = AB|C$ is in $\mathcal{R}(T, T)$ (hence, $\mathcal{R}(T, T) \neq \emptyset$).

The above properties can be formalized in different ways depending on the kind of topological relationship considered; e.g., clades, nestings, triplets, etc. Following Goloboff and Pol (2002) and Grunewald et al. (2006), we chose to focus on triplets. Given a collection $T$ of input trees and a candidate supertree $T$, $\mathcal{R}(T, T)$ denotes the set of triplets of $T$ for which $T$ proposes a resolution. More formally, $\mathcal{R}(T, T) = \{ AB|C \in rt(T) \text{ such that } AB|C, AC|B, BC|A \in rt(T) \neq \emptyset \}$. The set $\mathcal{R}(T, T)$ corresponds to all topological information present in collection $T$ that is related to the information present in supertree $T$. Using this notation, we can express the induction property (PI) and the non-contradiction property (PC) as follows:

- $T$ satisfies PI for $T$ if and only if for all $t \in rt(T)$, it holds that $\mathcal{R}(T, T) \vdash t$. In other words, PI requires that each and every triplet of $T$ is induced by $\mathcal{R}(T, T)$.
- $T$ satisfies PC for $T$ if and only if for all $t \in rt(T)$ and all $t'$, it holds that $\mathcal{R}(T, T) \not\vdash t$. This means that, for each and every triplet of $T$, $\mathcal{R}(T, T)$ induces no alternative resolution.

Here it is illustrated for another voting method, and one might wonder whether this size bias is present in most voting methods. Note, however, that this size bias does not seem to have a major impact on MRP's accuracy (Baum and Ragan, 2004).
When the source trees are compatible, any reasonable method is expected to produce a supertree satisfying PC. However, some methods usually propose a supertree that does not satisfy PI. Indeed, compatible source trees can sometimes be displayed by an exponential number of supertrees, and some methods arbitrarily propose only one of them, thus selecting some triplets to the exclusion of other possible triplets. For instance, when considering the trivial case of two source trees AB|E and CD|E, both MC and MMC propose the supertree ((A,B),(C,D),E) whereas numerous supertrees are possible; e.g., ((A,C),(B,D),E). In such a case, it seems preferable to output a consensus of all possible supertrees, as done by MRP (e.g., Bininda-Emonds and Bryant, 1998). Unfortunately, some topological information of the source trees (e.g., triplets) can be absent from the obtained consensus as it can contain highly multifurcating nodes.

However, for some compatible collections of trees, it is possible to find a supertree that displays all triplets of the collection and is also refined by all other possible supertrees. For instance, when considering the trivial case of two source trees AB|E and CD|E, both MC and MMC propose the supertree ((A,B),(C,D),E) whereas numerous supertrees are possible; e.g., ((A,C),(B,D),E). In such a case, it seems preferable to output a consensus of all possible supertrees, as done by MRP (e.g., Bininda-Emonds and Bryant, 1998). Unfortunately, some topological information of the source trees (e.g., triplets) can be absent from the obtained consensus as it can contain highly multifurcating nodes.

However, for some compatible collections of trees, it is possible to find a supertree that displays all triplets of the collection and is also refined by all other possible supertrees. More formally, a set \( R \) of triplets is said to identify a tree \( T \) if and only if \( T \) strictly displays \( R \) and \( T \) is refined by every tree \( T' \) that strictly displays \( R \). A set \( R \) can identify at most one tree; thus, when the triplet set \( R = rtt(T) \) of a collection \( T \) of source trees identifies a tree, this tree is a canonical representation of all possible supertrees.

Considering practical collections \( T \) of source trees, \( rtt(T) \) will almost never identify a tree, either because this set is incompatible or because it does not identify a particular tree. Nevertheless, it is possible that a subset of the triplets in \( rtt(T) \) identifies a tree \( T \), and then the topological information contained in \( T \) exactly corresponds to a subset of the topological information contained in \( T \). Such a subset is most interesting when the triplets \( I \) it contains do not have an alternative resolution \( I \) in \( rtt(T) \). This situation occurs for the subset \( R(T, T) \) of \( rtt(T) \) when the supertree \( T \) satisfies PI and PC.

Proposition 1 A tree \( T \) satisfies PI and PC for a collection \( T \) of trees if and only if \( R(T, T) \) identifies \( T \).

The proof is given in Appendix 1. It is based on the fact that a set \( R \) identifies a tree \( T \) if and only if \( rtt(T) = cl(R) \) (Grunewald et al., 2006: lemma 2.1). This proposition confirms the relevance of PI and PC: having a supertree \( T \) that satisfies both of them highlights a part of \( rtt(T) \) (namely \( R(T, T) \)) that exactly corresponds to a tree, i.e., does not contain arbitrary topological information, and moreover does not contradict any input tree. Such a feature is most desirable for supertrees inferred by veto methods.

Links with Other Advocated Properties

Properties similar to PI and PC were described in Goloboff and Pol (2002: 519) as "the property of [the supertree] displaying \( AB|C \) if it is found in some input tree or implied by some combination of input trees and no input tree or combination of input trees displays or implies \( AC|B \) or \( BC|A \)." These properties were also pointed out as being desirable by Grunewald et al. (2006). Using our formalism, they can be translated as follows for a supertree \( T \) representing a collection \( T \):

- \( PI' \): for any \( t \in rtt(T) \), it holds that \( rtt(T) \vdash t \)
- \( PC' \): for any \( t \in rtt(T) \) and for all \( I \), it holds that \( rtt(T) \nvdash I \).

The essential difference between \( PI' \)-\( PC' \) and PI-PC is whether we evaluate supertrees based on triplets in the original set of trees, \( rtt(T) \), or on the triplets commonly resolved by the supertree and at least one of the source trees, \( R(T, T) \). From the statement of the properties, it is clear that \( PC' \) implies \( PC \) and \( PI \) implies \( PI' \). It is thus natural to wonder which version of the properties is preferable. Below, we show an example where \( PC' \) is too restrictive, and an example where \( PI' \) is too permissive. In contrast, PI and PC behave correctly in these examples.
Example 1 Let $T = (T_1, T_2)$ with $T_1$ and $T_2$ as shown in Figure 4. $rt(T)$ contains $AE|B$ and $AC|E$; therefore, $rt(T) \vdash AC|B$. We also have $rt(T) \vdash AB|C$ because $AB|C \in rt(T_1)$. Thus any tree providing a triplet on $\{A, B, C\}$ does not satisfy $PC$. For analogous reasons $PC'$ does not allow us to propose any triplet in the supertree. Thus, $PC'$ rejects the tree $T$ of Figure 4. Yet $T$ is a reasonable and informative supertree for $T$ and satisfies both $PI$ and $PC$.

We note that $T$ is not a plenary supertree, i.e., it does not contain all input taxa, but this example shows that removing rogue taxa is a way in which more informative supertrees can be obtained. This is in line with the remark of Wilkinson et al. (2004), who stated that “non-plenary supertree methods might be most useful for identifying unstable leaves.” For instance, such leaves might be involved in lateral transfers. This example easily generalizes to cases where the supertree actually contains more leaves than each source tree. Figure 5 depicts this generalization.

The next example shows a supertree satisfying both $PI'$ and $PC'$ while also displaying irrelevant triplets.

Example 2 Let $T = (T_1, T_2)$ with $T_1$ and $T_2$ as illustrated in Figure 6. $rt(T) = \{AB|C, AB|X, BC|A\}$. The tree $T$ of Figure 6 displays $\{AB|X, BC|X, AC|X\}$. $AB|X$ is present in (thus induced by) $rt(T)$ but the two other triplets can also be induced from $rt(T)$: $\{AB|X, BC|A\} \vdash \{BC|X, AC|X\}$. It follows that $T$ satisfies $PI'$. Moreover, it is easily seen that no combination of triplets in $rt(T)$, other than $\{AB|X, BC|A\}$, induces triplets. However, $T$ is clearly not an ideal supertree for $T$ as no information in $T$ induces group $A, B, C$ to nest inside group $A, B, C, X$. The property $PI$, not satisfied by $T$, detects this problem: here $R(T, T)$ only contains the triplet $AB|X$ and thus it does not induce the triplet $AC|X$ present in $T$.

The $PI'$ property quoted by Goloboff and Pol (2002) is stronger than the Pareto property (Neumann, 1983; Wilkinson et al., 2004) on triplets, which requires that the output tree contain all triplets and splits that occur in all source trees. The Pareto property is appealing in general and has also been advocated in the supertree context (property P6 of Steel et al., 2000). However, imposing the Pareto property on triplets may be problematic, even in the case of compatible source trees (Thorley and Wilkinson, 2003). This is due to the possibility of having several candidate supertrees that are both compatible with source trees and respect the Pareto property. In this case, no single supertree exists that satisfies the Pareto property while having no arbitrary resolution. The strict consensus of these supertrees does not necessarily satisfy the Pareto property. A solution is then to return several trees, either all candidate supertrees or their reduced consensus (Wilkinson, 1994). However, this solution may not well be suited when the aim is to summarize a collection of source trees into a single supertree that is more easily dealt with for further analysis by biologists.

When source trees are incompatible, it may even be impossible to have a supertree satisfying both the Pareto and non-contradiction properties ($PC$ and $PC'$) as shown in the following example.

Example 3 Consider the collection $T = (T_1, T_2)$ where $T_1 = (((A, D), B), ((C, F), E))$ and $T_2 = (((A, E), (B, F)), (C, D))$. Triplets $AB|C$ and $EF|D$ are displayed by both trees of $T$. Thus, any supertree $T$ for $T$ must include all leaves in $T$ in order to satisfy the Pareto

![Figure 4](image-url) Excluding rogue taxa from the analysis can lead to informative supertrees.

![Figure 5](image-url) Another example with rogue taxa. This figure presents a generalization of the example displayed in Fig. 4 to the case of a supertree containing more taxa than each input tree.
property. Since \( rT \) contains \( AB|D \) and \( AD|B \), any tree \( T \) displaying a triplet for the three leaves does not satisfy PC (hence PC'). For similar reasons, no supertree \( T \) can display a triplet on the taxa \( A, C, \) and \( D \). Thus, any supertree satisfying PC (or PC') and including all taxa of \( T \) contains a multifurcating node on taxa \( A, B, C, \) and \( D \), and hence does not display the triplet \( AB|C \); i.e., does not satisfy the Pareto property.

In other words, imposing the Pareto property can lead the supertree to explicitly contradict relationships present in some input trees. This shows that the Pareto property on triplets is not compatible with the veto approach, where the proposed supertree must not contradict the source trees. However, the Pareto property can be considered for other topological relationships (Wilkinson et al., 2004). For example, there is always a supertree satisfying PI and PC as well as the Pareto property on partial or full splits contained in the source trees.

The Pareto property specifies relations that the supertree must contain. The complementary co-Pareto property specifies relations that the supertree must not contain. The co-Pareto property in the consensus context requires that the consensus tree contain no relationships that are not present in at least one input tree. However, Wilkinson et al. (2004) point out that this statement is not reasonable for supertrees, because “they might contain relationships that are entailed by the input trees in combination, but are not present in any of them singly.” Then they propose a weaker version that requires that the supertree does not contain relationships that are contradicted by all the input trees whose leaf set makes a contradiction possible. Note that, any supertree satisfying PC also satisfies the latter version of co-Pareto.

Steel et al. (2000) list five other properties that might be requested from supertree methods: changing the order of the trees in the input collection does not change the supertree (P1); renaming the taxa of the source trees gives the same supertree, but with the taxa renamed accordingly (P2); the output tree displays the source trees when they are compatible (P3); each leaf (taxon) that occurs in at least one source tree is in the supertree (P4); the running time of the method grows polynomially with respect to the total number of taxa (P5). The following example shows that ensuring P3 can force the supertree to contain arbitrary clades. Thus, P3 can conflict with PI.

**Example 4** Let \( T = \{T_1, T_2\} \) with \( T_1 = ((A, B), W) \) and \( T_2 = ((A, B), (X, Y, Z)) \). A supertree with taxon set \{A, B, W, X, Y, Z\} that satisfies P3 must display \( T_2 \), hence must have a clade including \( Y, Z \) but not \( X \). However, it will contain arbitrary clades, no matter where taxon \( W \) is attached. This is because any supertree satisfying PI must include a polytomy on \( W, X, Y, Z \) since source trees include no information on the relative position of \( W \) and the group \( X, Y, Z \). For instance, the supertree \((A, B), (X, Y), W, Z\) excludes the possibility for \((A, B)\) and \((X, Y)\) to be intermixed.

Note that if polytomies of a supertree are interpreted in terms of an Adams consensus (Adams, 1972), then this example does not put P3 into question. However, this interpretation of polytomies does not prevail in phylogenetics, as we discuss in further detail in the case study paragraph.

**Checking PC and PI in Polynomial Time**

Existing supertree methods can sometimes output trees that do not satisfy PC or PI. For instance, the MC supertree obtained for the collection of Figure 2 does not satisfy PC, whereas on that of Example 4, it fails to satisfy PI. In contrast, for the collection \{AB|C, BC|D\} the MC supertree satisfies both PI and PC. The MRP method sometimes outputs supertrees not satisfying these properties (e.g., PC is not satisfied in Figure 1 of Bininda-Emonds and Bryant, 1998) and sometimes provides supertrees that satisfy them—e.g., when the source trees are compatible (Steel, 1992). We now describe an algorithm to decide whether a candidate supertree satisfies both PI and PC together. In case of a negative answer, it pinpoints those parts of the supertree contradicting these properties. This algorithm relies on two properties equivalent to PC and PI, whose formulation is less intuitive but whose checking is easy.

**Definition 1** Let \( T \) be the collection of source trees and \( T \) be a proposed supertree for \( T \). Define \( PC_eq \) and \( PI_eq \) to be the following properties:

- \( PC_eq \): \( rT \cap R(T, T) \) is compatible.
- \( PI_eq \): for any \( t \in rT \) and for all \( I \), the set \( \{I\} \cup R(T, T) \) is incompatible.

**Proposition 2** \((PI_eq \text{ and } PC_eq) \Rightarrow (PI \text{ and } PC)\).
Formulation of PC, closer to that of PI, but less compatible.

However, it is possible to check this condition in linear time for each pair T, T in T, with T, T being the first restriction in O(n) time the trees T and T to the taxa they share; then apply the algorithm of Berry et al. (2005) that, given two trees with the same taxa, finds in O(n) time a triplet resolved differently in the trees, or states that this situation does not arise. Thus, successively considering k source trees leads to a procedure that checks PC in O(kn) computing time.

**PhySIC: A Polynomial-Time Veto Supertree Method**

We introduced above the PI and PC properties, showed their relevance, and described algorithms to check whether a given supertree T satisfies them. In this section, we show that it is possible to design a method that always produces supertrees that satisfy PI and PC. However, this aim is not precise enough, as the star tree (the tree whose leaves are all children of a single internal node) trivially satisfies these properties—simply because it does not resolve any triplet. Thus, a reasonable aim is to design a method that always infers supertrees that satisfy PI and PC and that contain as much resolution as possible; e.g., resolve as many triplets as possible. More precisely, we require a method that, given any collection T, proposes a supertree T such that R(T, T) identifies T and R(T, T) has maximum size over all such subsets of r(T). Such a subset of r(T) is called a maximum identifying subset of triplets (MIST).

The difficulty of this problem cannot be simply deduced from previously known theoretical results for optimization problems on triplets. Indeed, the MIST problem is a middle term between the NP-hard problem that consists of finding a maximum-sized compatible subset of triplets (Bryant, 1997) and the polynomial-time problem that asks for the maximum-sized tree-like subset of a complete set of triplets (Bryant and Gascuel, 2000; Bryant and Berry, 2001). Unfortunately, the MIST problem is NP-hard (Guillemot and Berry, 2007). This shows that it is highly unlikely that a polynomial-time algorithm exists that could find the most resolved supertree satisfying PI and PC. However, we can still rely on heuristic algorithms to find reasonable (but potentially suboptimal) solutions, as is commonly done for other NP-hard problems such as finding a most parsimonious tree or a maximum likelihood tree for a character matrix.

We present below a polynomial-time heuristic method that always outputs a supertree that satisfies PI and PC. The method tries to produce a supertree that contains as many input triplets as possible under this constraint. The method is a variant of the well-known Build algorithm.
and is called \textit{PhySIC—Phylogenetic Signal with Induction and non-Contradiction}. Supertrees inferred by the method have a degree of resolution that can be close to that of supertrees inferred by voting methods (see next section) while only containing clades that are not arbitrary with respect to the source trees nor contradicting them as detected by PC.

\textbf{Inferring a Supertree that Satisfies PC}

This section introduces algorithms, based on the \textit{Build} algorithm, to produce nontrivial trees that satisfy PC.

\textit{The Build Algorithm}.—The \textit{Build} algorithm is a yes-or-no algorithm that tells whether a collection of triplets or larger trees is compatible or not. To achieve its goal, the algorithm tries to build a tree displaying the triplets; if the process is blocked at some step, this means that the input triplets are not compatible. This tree is built recursively, from the root to the leaves. First, the largest clades are identified, then clades included in the first ones, and so on. The composition of the clades is guided by the structure of a graph; that is, a set of objects (called vertices) with links (called edges) between pairs of them.

The graph used by \textit{Build}, called here the Aho Graph, is defined as follows: let \( R \) be a collection of triplets on a taxon set \( X \), the Aho Graph \( G \) for \( R \) is the undirected graph with vertices \( X \) and with edges \((A, B)\) between two taxa \( A \) and \( B \) whenever there is a triplet \( AB \in R \). Thus, an edge between two taxa means that at least one triplet sees these two taxa in the same clade. The vertices of \( G \) are denoted by \( v(G) \) (in the present description \( v(G) = X \)). For instance, Figure 7a shows the Aho graph built from \( R = r(t(T_1, T_2)) \), where \( T_1, T_2 \) are the source trees of Figure 2 and, e.g., the edge between taxa \( B \) and \( D \) is due to the triplet \( bd\in C_2 \).

A connected component \( C_i \) of a graph is a maximal set of taxa linked to one another; i.e., such that for any pair \( A, B \) of taxa in \( C_i \), there is a set of edges that links \( A \) to \( B \). For instance, the graph in Figure 7a contains two connected components: \( C_1 = \{E, F\} \) and \( C_2 = \{A, B, C, D\} \). The connected components of graph \( G \) are denoted by \( CC(G) \). The vertices of a component \( C_i \) of \( G \) are denoted by \( v(C_i) \). When the Aho graph contains several connected components, each of them corresponds to a clade of the tree representing the input collection of triplets (if such a tree exists). Once these clades are known, the clades contained in each of these primary clades are found by recursively processing Aho graphs for subsets of triplets that respectively concern the taxa of these clades: the restriction of \( R \) to taxa of a component \( C_i \) is denoted by \( R_{\mid v(C_i)} \) and defined as \( \{AB\mid C \in R \text{ such that } \{A, B, C\} \subseteq v(C_i)\} \). For example, Figure 7b shows the Aho graph obtained from \( R_{\mid v(C_2)} \) where \( R \) is the set of triplets due to source trees in Figure 2, and \( C_2 \) is the component of the initial Aho graph shown in Figure 7a. The recursive calls stop when dealing with components containing less than 3 taxa, because there is no triplet (hence incompatibility) on so few taxa. However, if at some point in the recursive process, the Aho graph for a set of at least three taxa has only one connected component, this means that the input trees are conflicting on the resolution of these taxa. When this happens, the algorithm states that the collection of source trees is incompatible. Otherwise, when all recursive calls return, the algorithm concludes that the source trees are compatible. For instance, when run on the collection of Figure 2, \textit{Build} first finds two connected components, \( C_1 = \{E, F\} \) and \( C_2 = \{A, B, C, D\} \), but the recursive call on \( C_2 \) leads to a graph containing only one connected component (Figure 7b), which leads the algorithm to detect the incompatibility of the source trees.

\textit{A First Simple Modification of Build}.—We first describe here a simple modification of \textit{Build} that infers a supertree from a collection of source trees \( T \). This subroutine, called \textit{BuildPC} (Figure 8), takes as input the triplet set \( R = r(t(T)) \) of a collection \( T \) of source trees and the list \( S \) of taxa contained in these trees. \textit{BuildPC} mainly differs from \textit{Build} when the Aho graph contains one connected component on the set \( S \) of taxa currently considered (line 1). In this case, \textit{BuildPC} returns the star tree on \( S \) (i.e., a single polytomy on \( S \), thus contradicting no input triplet), whereas \textit{Build} simply concludes that the sources trees are incompatible. This star tree is then graftied as a sub-tree of the tree built by the previous recursive call. Thus, we can now output a supertree even when the source trees are incompatible. As an example, from the collection of Figure 2, \textit{BuildPC} infers the supertree \( T' \) displayed in Figure 3.

\textbf{Proposition 3} Given a collection \( R \) of triplets on a taxon set \( S \), \textit{BuildPC} returns a tree \( T \) on \( S \) that satisfies the PC property for \( R \).

The proof of this proposition can be found in Appendix 1.

\textit{A More Involved Algorithm to Infer a Supertree Satisfying PC}.—\textit{BuildPC} sometimes produces poorly resolved trees due to multifurcations returned in cases where \( G \) contains a single connected component (i.e., when \( R \) contains conflicts covering the considered subset of taxa). In the most extreme (though unlikely) case, this situation occurs at the first step of the algorithm, which then outputs a star tree.

The most basic conflicts between triplets of \( R \) occur when two different triplets \( t \) and \( I \) appear in \( R \) for a same set of three taxa. Such a direct contradiction cannot be present in a tree that satisfies PC. Given \( R_{dc} \), the set of triplets such that \( t, I \in R \) it seems relevant to consider the subset \( R' = R - R_{dc} \). We define a variant of \textit{BuildPC}, called \textit{PhysSICPC}, that resorts to that subset whenever conflicts are detected. This enables the produced supertree \( T' \) to be generally much more resolved than the tree returned by \textit{BuildPC}. For instance, Figure 7b shows the graph obtained for \( R_{\mid v(C_2)} \), where \( R \) are triplets of the collection in Figure 2 and \( C_2 \) is the connected component shown in Figure 7a. This graph is connected due to the direct conflicts between \( AB\mid C \) (resulting from \( T_1 \)) and \( BC\mid A \) (resulting from \( T_2 \)). This situation leads \textit{BuildPC} to return a polytomy on \( A, B, C, D \). In contrast, building the graph on the basis of \( R' \) results in two connected
2007 RANWEZ ET AL.—DESIRABLE PROPERTIES FOR VETO SUPERTREES

807

FIGURE 7. Examples of graphs. (a) The initial Aho graph G created from the triplets \( r(T) \) of the collection \( T \) displayed in Figure 2. The two connected components of G are \( C_1 = \{E, F\} \) and \( C_2 = \{A, B, C, D\} \). (b) The Aho graph obtained from \( \mathcal{R}(\mathcal{A}(C_2)) \). This graph is connected, showing that the input trees conflict on the resolution of \( \{A, B, C, D\} \) and hence are incompatible. (c) The Aho graph obtained from \( \mathcal{R}(\mathcal{B}(C_2)) \) when removing the triplets \( \mathcal{R}_{dc} = \{AB|C, AC|B\} \).

components, \( C_1 \) and \( C_2 \), allowing \( \text{PhysIC}_{PC} \) to propose a tree with two subtrees for taxa \( A, B, C, D \).

The correctness of \( \text{Build}_{PC} \) ensures that \( T' \) satisfies PC with respect to \( \mathcal{R}' \) but without any guarantee that this also holds with respect to \( \mathcal{R} \). To ensure the latter, and thus the correctness of \( \text{PhysIC}_{PC} \), \( T' \) must not resolve any triplet of \( \mathcal{R}_{dc} \). A way to ensure this is to collapse any branch of \( T' \) that resolves a triplet of \( \mathcal{R}_{dc} \). The tree thus obtained is still always at least as resolved as the one proposed by \( \text{Build}_{PC} \) and potentially contains supplementary branches. Indeed, direct contradictions at the root of a clade no longer prevent the proposition of clades on subsets of its taxa. For instance, on the collection of Figure 2, the tree initially computed by \( \text{PhysIC}_{PC} \) is the tree called \( T \) in Figure 3. But as the branch leading to the clade \( (A, D, B) \) contradicts \( AC|B \in \mathcal{R}_{dc} \), the branch above this clade is collapsed, and the final tree output by \( \text{PhysIC}_{PC} \) is then the tree named \( T'' \) in Figure 3. This tree contains one clade more than the tree output by \( \text{Build}_{PC} \) (the tree named \( T' \) in the figure).

These ideas are included in the \( \text{PhysIC}_{PC} \) algorithm whose pseudo-code is detailed in Figure 9.

Theorem 1 Given a triplet set \( \mathcal{R} = r(T) \) from a collection \( T \) of source trees on a taxon set \( S \) of \( n \) leaves, \( \text{PhysIC}_{PC} \) returns in \( O(n^4) \) time a tree \( T \) satisfying PC for \( T \).

Algorithm \( \text{Build}_{PC}(S, \mathcal{R}) \)

\[
\begin{array}{l}
\text{if} \ S \ \text{contains less than 3 taxa then} \ \text{return the trivial tree on } S \\
\text{Let } G \text{ denote the Aho graph for } \mathcal{R} \\
\text{if} \ G \text{ has only one connected component then} \\
\quad \text{return the star tree on } L(\mathcal{R}) \\
\text{else} \\
\quad C_{PC} \leftarrow CC(G) \\
\quad \text{foreach } C_i \in C_{PC} \text{ do} \\
\quad \quad \text{if } (\mathcal{R}|v(C_i)) = \emptyset \ \text{then} \ T_i \leftarrow \text{star tree on } v(C_i) \\
\quad \quad \quad \text{else} \ T_i \leftarrow \text{Build}_{PC}(v(C_i), \mathcal{R}|v(C_i)) \\
\quad \text{Return the tree made of a root node connected to } T_1, T_2, ..., T_{|C_{PC}|} \\
\end{array}
\]

FIGURE 8. Details of the \( \text{Build}_{PC} \) subroutine taking a set \( S \) of taxa and a set \( \mathcal{R} \) of triplets on \( S \) as input.
Algorithm $\text{PhySIC}_{PC}(S, \mathcal{R})$

if $S$ contains less than 3 taxa then return the trivial tree on $S$
Let $G$ denote the Aho graph for $\mathcal{R}$
if $G$ has several connected components then $C_{PC} \leftarrow CC(G)$
else
   Let $R_{dc}$ be the set of triplets $t$ s.t. $t, \bar{t} \in \mathcal{R}$
   $\mathcal{R}' \leftarrow \mathcal{R} - R_{dc}$
   Let $G'$ be the Aho graph for $\mathcal{R}'$
   if $G'$ is connected then $C_{PC} \leftarrow v(G)$
   else
      $C_{PC} \leftarrow CC(G')$
   repeat
   foreach $AB \in \mathcal{R}_{dc}$ do
      if $A, B \in C_i$ and $C \in C_j$ (with $C_i, C_j \in C_{PC}$ and $i \neq j$) then
         Build $G'_i$ the Aho graph for $\mathcal{R}'|v(C_i)$
         if $G'_i$ is connected then $C_{PC} \leftarrow (C_{PC} - \{C_i\}) \cup v(C_i)$
         else $C_{PC} \leftarrow (C_{PC} - \{C_i\}) \cup CC(G'_i)$
   until $C_{PC}$ no longer changes
   foreach $C_i \in C_{PC}$ do
      if $|\mathcal{R}'|v(C_i)) = 0$ then $T_i \leftarrow$ star tree on $v(C_i)$
      else $T_i \leftarrow $ PhySIC$_{PC}(v(C_i), \mathcal{R}|v(C_i))$
   Return the tree made of a root node connected to $T_1, T_2, ..., T_{|C_{PC}|}$

**Figure 9.** Details of the PhySIC$_{PC}$ subroutine taking a set $S$ of taxa and a set $\mathcal{R}$ of triplets on $S$ as input.

This theorem is the basis of a decision algorithm called Identifies that states whether a given set of triplets identifies a given tree (Daniel, 2004). It is possible to design a simple variant of this algorithm that always returns a tree (not just a yes or no answer): when a branch between a node $p$ and the root of a subtree $S_i$ is not justified, the idea is to replace $S_i$ by a star tree on the taxa of the corresponding clade. This crude variant removes the unjustified branches, but also potentially many other branches; i.e., those inside $S_i$, those leading to sibling subtrees $S_j$ of $S_i$, and those inside $S_j$ subtrees. PhySIC$_{PI}$ is a more refined variant that only collapses the unjustified branches. See the pseudo-code in Figure 10 for details. In this code, PhySIC$_{PI}$ is given a tree $T$ in which unjustified branches are to be collapsed and a collection $T'$ of source trees or, equivalently, the corresponding set of triplets (as written in the pseudo-code). PhySIC$_{PI}$ repeatedly calls the Check$_{PI}$ subroutine to detect unjustified branches that are then removed until none remains (note that in the pseudo-code of Check$_{PI}$, $S(T)$ denotes (complete) subtrees connected to the root of $T$; i.e., the subtrees corresponding to the largest clades under the root of $T$).

From the collection of Figure 2, PhySIC$_{PC}$ infers the supertree $T''$ displayed in Figure 3 and none of the three internal branches of $T''$ are collapsed by Check$_{PI}$. For instance, consider the step where Check$_{PI}$ checks the subtree $((A,D),B,C)$ of $T''$, whose child subtrees are $(A,D)$ plus the two trivial subtrees on $B$ and $C$. The sole branch that has to be checked in $((A,D),B,C)$ is the one defining the clade $(A,D)$. Here, Check$_{PI}$ builds two Aho graphs with vertices $(A, D)$: one with edges due to triplets on $(A, D) \cup \{B\}$ and one with edges due to triplets on $(A, D) \cup \{C\}$. Both graphs are connected thanks to triplets of the source tree $T_2$; therefore, Check$_{PI}$ does not collapse any branch at this step.

**Theorem 2** Given a collection $T$ of trees and a tree satisfying PC for $T$, PhySIC$_{PI}$ returns in $O(n^4)$ time a tree $T$ on $L(T)$ that satisfies both PC and PI for $T$.

The proof of this Theorem can be found in the appendix.
Algorithm \textit{PhySIC}$_{P1}$ ($T$)
\begin{align*}
& T_{P1} \leftarrow T \\
& \text{repeat} \\
& \quad \mathcal{R}_{P1} \leftarrow \mathcal{R}(T_{P1}, T) \\
& \quad T_{P1} \leftarrow \text{Check}_{P1} (T_{P1}, \mathcal{R}_{P1}) \\
& \text{until } T_{P1} \text{ no longer changes} \\
& \text{Return } T_{P1}
\end{align*}

Algorithm \textit{PhySIC} ($T$)
\begin{align*}
& \text{Let } S \text{ be the taxa appearing in } T \\
& \quad \mathcal{R} \leftarrow rt(T) \\
& \quad T_{PC} \leftarrow \text{PhySIC}_{PC} (S, \mathcal{R}) \\
& \quad \text{Return } \text{PhySIC}_{P1} (T_{PC}, \mathcal{R})
\end{align*}

Algorithm \textit{Check}_{P1} ($T$, $\mathcal{R}$)
\begin{align*}
& \text{if } T \text{ is made of a single leaf then return } T \\
& \text{Let } G \text{ be the Aho graph for } \mathcal{R} \\
& \frac{}{12} \text{if } |CC(G)| = 1 \text{ then return } \text{"error, } \mathcal{R} \text{ is incompatible"} \\
& \text{repeat} \\
& \quad \text{foreach } T_i \in S(T) \text{ do} \\
& \quad \quad \text{Let } G_i \text{ be the Aho graph for } \mathcal{R}|L(T_i) \\
& \quad \quad \text{foreach } T_j \in S(T) \text{ s.t. } T_i \neq T_j \text{ do} \\
& \quad \quad \quad \text{Build } G_{ij} \text{ from } G_i \text{ and } \mathcal{R}|(L(T_i) \cup L(T_j)) \\
& \quad \quad \quad \text{if } G_{ij} \text{ is not connected then} \\
& \quad \quad \quad \quad \text{Collapse the branch between the root} \\
& \quad \quad \quad \quad \text{of } T \text{ and } T_i \\
& \text{until no branch of } T \text{ is collapsed} \\
& \text{foreach } T_i \in S(T) \text{ do} \\
& \quad \text{Let } G_i \text{ be the Aho graph for } \mathcal{R}|L(T_i) \\
& \quad \text{Return the tree made of a root node connected to} \\
& \quad T'_1, T'_2, \ldots, T'_{|S(T)|}
\end{align*}

The \textit{PhySIC} algorithm (see pseudo-code) builds a supertree for a collection of \( k \) source trees \( T \) by first computing the set \( rt(T) \) and then successively calling \textit{PhySIC}_{PC} and \textit{PhySIC}_{P1}. Since \( rt(T) \) is computed in \( O(kn^3) \), \textit{PhySIC} runs in \( O(kn^3 + n^4) \) time.

\textbf{Theorem 3} \textit{Given a collection of } \( k \text{ source trees on } n \text{ leaves, } \textit{PhySIC} \text{ returns in } O(kn^3 + n^4) \text{ time a tree satisfying both } PC \text{ and } PI. \}

Lastly, we note that similar procedures can be designed to modify the supertree proposed by any existing supertree method. If a method proposes a supertree \( T \) that does not satisfy PC and PI, it is possible in polynomial time to transform \( T \) into a tree \( T' \) that satisfies these properties. Indeed, the algorithm indicated previously to check PC indicates the triplets from which the incompatibility arises. Then the branches of \( T \) inducing these triplets can be collapsed to obtain a tree \( T' \) satisfying PC. Now, \textit{PhySIC}, can be applied to \( T' \) to ensure that it also satisfies PI (without invalidating PC).

\textbf{BIOLOGICAL CASE STUDIES ON PRIMATES}

To illustrate the impact of the PC and PI properties on supertree inference, and to compare the behavior of veto methods like \textit{PhySIC} to that of voting methods like MRP and MMC, we present two case studies centered on Primates. This mammalian order is one of the first taxonomic groups for which a large-scale supertree approach has been conducted (Purvis, 1995a). The first example is designed to show the desirable properties of \textit{PhySIC} compared to other supertree methods on a smaller, understandable taxonomic scale. The second example addressing the question of the primate supertree at the genus level shows how \textit{PhySIC} performs on a larger taxonomic scale—approaching what supertree studies tend to be performed on—and shows that varying degrees of resolution are achieved in the supertree depending upon the nodes retained from the input trees.

\textit{First Example: Illustration of Supertree Desirable Properties}

\textbf{Source Trees.}—We focused on a subsample of Primates IRBP (interphotoreceptor retinoid binding protein) and ADRA2B (\( \alpha_2B \)-adrenergic receptor) gene sequences, respectively, from Poux and Douzery (2004) and Poux et al. (2006), with a rodent (\textit{Mus}) and lagomorph (\textit{Oryctolagus}) outgroup. For ADRA2B, the hominoid representative was \textit{Pan}, with the sequence downloaded from the chimp ENSEMBL project. The ADRA2B and IRBP source trees were inferred by maximum likelihood (ML) analysis of the corresponding alignments, using PHYML (Guindon and Gascuel, 2003), version 2.4.4, under a GTR+\( \Gamma_4 \)+INV model of DNA evolution. The node support was estimated after 1000 bootstrap replicates using the same software and expressed as bootstrap percentages (BP). Denser taxonomic and phylogenetic information for Strepsirrhines (i.e., \textit{Lemurs} and \textit{Galagos}) was sought from
a study of presence-absence of short interspersed nuclear elements (SINE) integrations in primate genomes (Roos et al., 2004: fig. 2). Sixty-one monocus SINE characters detected by these authors were subjected to a maximum parsimony analysis using PAUP* (Swofford, 2002), version 4b10, with 1000 bootstrap replicates using a heuristic search, with 10 random additions of taxa, and TBR branch swapping. We only retained the best supported nodes of source trees; i.e., those showing at least 50% bootstrap (cf. also Daubin et al., 2002).

Comparison of Supertrees Inferred from PhySIC, MMC, and MRP.—Starting from the three source trees (Figure 11), supertrees were built using the MMC, MRP, and PhySIC methods. For MRP, the matrix representation of the three source topologies resulted in 47 characters. Parsimony analysis was conducted under PAUP*, with a heuristic search with 1000 random addition sequences, and TBR branch swapping, resulting in 864 equally parsimonious trees, a strict consensus of which provided the MRP supertree. The MMC supertree was obtained using the program distributed by Rod Page. Figure 12 shows the supertrees respectively reconstructed by MMC, MRP, and PhySIC with its PhySIC \textit{PC} intermediate step.

The supertrees produced all contain some soft polytomies, each of them representing uncertainty about the resolution of a node’s child subtrees or lineages. A soft polytomy can have two distinct interpretations, differing in the set of admissible fully resolved phylogenies it encompasses. Consider the case of the polytomy node $P$ in the MMC tree of Figure 12. This node has three child subtrees $S_1 = (\text{Homo}, \text{Hylobates})$, $S_2 = (\text{Pan}, (\text{Cercopithecus}, \text{Macaca}))$, and $S_3$, the Platyrhini clade. The most widespread meaning of a soft polytomy accepts any fully resolved tree on subtrees $S_1$, $S_2$, $S_3$ that keeps their monophyly: $((S_1, S_2), S_3)$, $((S_1, S_3), S_2)$, or $((S_2, S_3), S_1)$. Strict consensus, majority-rule consensus, and hence MRP, interpret polytomies in this way (Margush and McMorris, 1981). Polytomies proposed by PhySIC \textit{PC} are also to be interpreted in this way. A second interpretation of soft polytomies was introduced by the Adams consensus (Adams, 1972) and is also intended by MC (Semple and Steel, 2000) and MMC (Page, 2002). This interpretation accepts as possible phylogeny any fully resolved tree that maintains the structure of each subtree respectively, no matter whether or not $S_1$, $S_2$, and/or $S_3$ are kept monophyletic (i.e., their leaves can be interleaved). Thus, the polytomy $P$ of the MMC tree in Figure 12 can indeed give rise to fully resolved trees grouping $\text{Pan}$ and $\text{Homo}$ without $\text{Hylobates}$, as long as $\text{Pan}$ is kept outside the clades containing $\text{Cercopithecus}$ and $\text{Macaca}$ (which is the structure imposed by $S_1$). Under this interpretation, a soft polytomy represents a much wider range of fully resolved phylogenies than with the first interpretation, and is harder to interpret in a phylogenetic context. (In particular, this means that simulation studies on supertree methods that use the Robinson and Foulds distance to evaluate the performance of MC or MMC are misleading: on the previous example, the MMC method would have been considered to propose the incorrect clades $\text{Homo} + \text{Hylobates}$, and $\text{Pan} + \text{Hylobates}$.)

The contribution of PhySIC \textit{PC} to the supertree inference may be illustrated by the situation among platyrhines. Here, ADRA2B indicates that $\text{Ateles}$ is the

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure11.png}
\caption{Three source trees for the case study on primates. Majority rule consensus source trees derived from the bootstrap analysis of ADRA2B and IRBP sequences (in maximum likelihood) and SINE characters (in maximum parsimony). Bootstrap percentages are indicated on nodes, and only nodes defined by more than 50% are considered. Thin branches lead to the outgroup. Taxa in bold are handled differently by the different supertree algorithms and illustrate three different situations (arrows, letters, box: see main text). The two \textit{Tarsius} species are \textit{T. bancanus} (b) and \textit{T. syrichta} (s).}
\end{figure}
Figure 12. Comparison of supertrees inferred by three methods: MMC, MRP, and PhySIC (the PhySIC_{PC} intermediate step of the latter is also displayed). Thin branches lead to the outgroup. Taxa in bold are handled differently by the different supertree algorithms and illustrate three different situations. (i) Arrows indicate the surprising positioning of Homo and Pan under the MMC and PhySIC_{PC} algorithms; (ii) A-B-C-X letters correspond to taxa arbitrarily grouped by MMC and PhySIC_{PC} (cf. Figure 2); (iii) boxes contain platyrrhine taxa for which MMC and MRP contradict the IRBP source topology. The P label on MMC and PhySIC_{PC} supertrees refers to the polytomy involving catarrhine and platyrrhine clades. The taxonomic frame for Primates is given on the PhySIC supertree. Hatched rectangles represent Anthropoidea (Catarrhini + Platyrrhini). White and black rectangles respectively represent Haplorrhini (Tarsiiformes + Anthropoidea) and Strepsirrhini (Lorisiformes + Lemuriformes).
sister-group of *Pithecia*, *Callithrix*, and *Cebus*, whereas IRBP indicates that *Ateles* and *Cebus* are the closest relatives (Figure 11: boxed areas). This conflict is detected by *PhySICpc*. As a result, the *PhySICpc* and *PhySIC* supertrees display all four platyrrhines within a multifurcation. By contrast, MRP and MMC give priority to the *Callithrix* + *Cebus* grouping present in the ADRA2B source tree, and thus contradicts the *Ateles* + *Cebus* grouping present in the IRBP source tree. Resolution of this conflict between the source trees reflects the voting approach followed by MRP and MMC. For instance, consider the case of MRP: the ADRA2B source topology comprises two nodes within platyrrhines, against one node for the IRBP topology. Therefore, MRP favors the node *Callithrix* + *Cebus* involved in a topological conflict but belonging to a larger and more resolved clade (Bininda-Emonds and Bryant, 1998). The behavior of MRP and MMC on platyrrhines is problematic. Indeed, it favors one source topology while contradicting another, just on the basis of their respective levels of resolution, and despite the fact that both contain the same number of taxa for the platyrrhine subtree. MRP has already been criticized on this point (e.g., Goloboff, 2005). Note that source trees also conflict on the position of *Protopithecus* with respect to *Microcebus* and *Lemur*. However, in this case, MRP behaves as *PhySICpc* and *PhySIC*; i.e., displays a polytomy on groups containing these three taxa (Figure 12: letters A-B-C). By contrast, MMC groups together the *Protopithecus* and *Lemur* clades, following the SINE information, but contradicting the IRBP information.

The complementary contribution of *PhySICpc* to the supertree inference may be illustrated by the situation among Catarrhines + Platyrrhines. Although not contradicting the source trees, the *PhySICpc* supertree contains two topological errors. First, man and chimp do not group together relative to the gibbon, as would be expected from a plethora of data (Goodman et al., 2005). *Homo* is instead associated with *Hylobates*, whereas *Pan* branches with the two cercopithecoids, *Cercopithecus* and *Macaca*. This situation results from the taxon sampling of the source topologies. More precisely, man and chimp are not simultaneously present in any source tree; i.e., the former clusters with the gibbon (IRBP) and the latter with cercopithecoids (ADRA2B). These two source clades are reproduced in *PhySICpc* and MMC supertrees.

In the case of *PhySICpc*, these two clades are involved in a polytomy with the platyrrhines. This polytomy means that (*Homo*, *Hylobates*) is a sister clade of the clade containing *Pan*. However, although these clades are correct when considered separately, they should not be sister groups in the supertree. *PhySICpc* detects this situation of arbitrary resolution and collapses the corresponding branches, thus the final *PhySIC* supertree allows for a group (*Pan*, *Homo*). MMC displays the same polytomy as *PhySICpc* but with a different meaning: the interleaving interpretation of this soft polytomy means that MMC does not reject the expected resolution, namely grouping (*Pan*, *Homo*) as a sister clade of *Hylobates*. In conclusion, both MMC and *PhySIC* allows for the expected group (*Pan*, *Homo*), but note that the *PhySIC* supertree is more accurate, as its polytomy does not allow the catarrhine taxa *Homo*, *Hylobates*, or *Pan* to branch within the platyrrhines. Here, MRP does not introduce arbitrary resolutions and proposes a polytomy involving the five catarrhine taxa.

Another problem of MMC and *PhySICpc* supertrees is that *Lepilemur* is the sister-group of all Lemuriformes but *Daubentonia*, whereas this topological information is not present in the only source tree (SINEs) for which *Lepilemur* is scored. This result is explained by the fact that the restriction of IRBP and ADRA2B source topologies to taxa lettered A-B-C-X leads to the situation described on Figure 6. Thanks to the PI property, the *PhySIC* algorithm again corrects this problem and displays a polytomy involving the major clades of lemuriformes, together with *Lepilemur* (Figure 12). The same polytomy is also proposed by MRP. Overall, this first case study illustrates that the two properties introduced in the present work help to identify and manage the potential arbitrary and conflicting resolutions arising in supertrees when combining independent source topologies.

**Second Example: A PhySIC Supertree of Primate Genera**

**Primary Data and Source Tree Inference.**—We used 24 data sets to reconstruct the primate phylogeny: two mitochondrial DNA (mtDNA), 19 nuclear DNA, and three transposable elements data sets. All sequences used in this study were retrieved from EMBL-Genbank databases. The sampling of genes and other molecular markers is detailed in Table 1. The corresponding data are available under TreeBASE accession numbers S1879 and M3455. This combined data set encompasses 95% of all primate extant genera (Wilson and Reeder, 2005); i.e., 66 genera. Two subfossil genera from ancient DNA analyses were also included (Karanth et al., 2005). All genes were aligned with Clustal X (Thompson et al., 1997) with subsequent manual refinement. We used *Mus* and *Rattus* as outgroups in all analyses for which sequence data was available. Each gene was analyzed with the ML criterion under the best fitting model (Table 1). Separate ML phylogenetic reconstructions and bootstrap analyses were performed with PHYML (Guindon and Gascuel, 2003) as described in previous section. Maximum parsimony phylogenetic reconstruction and bootstrap analysis on the three transposable element data sets were also conducted using PAUP* as described in previous section. Clades of the source trees with BP values above a specified threshold were retained. To evaluate the influence of this parameter, five *PhySIC* supertrees were inferred by respectively considering BP ≥ 50%, 60%, 70%, 80%, and 90%. Each run of *PhySIC* took less than 4 s on an Intel MacBook.

**The Major Clades of Primate Genera.**—The most resolved supertree reconstructed by *PhySIC* is obtained when source trees were restricted to nodes supported by more than 70% bootstrap (Figure 13; BP ≥ 70%). This topology conforms to current ideas on primate phylogeny and is close to the informal supertree of Primates at the genus level proposed by Goodman et al. (2005). In
addition, we here extend their taxon sampling with the three extant genera *Euticus, Pilocolobus,* and *Simias,* and the two subfossil genera *Megaladapis* and *Paleopropithecus.* Our supertree displays the fundamental dichotomy among Primates between Strepsirhini and Haplorrhini. Strepsirrhines then split into Lorisiformes (Lorisises and Callagros) and Lemuriformes (lemurs and Daubentonia, the aye-aye). Haplorrhines also split into Tarsiids and Anthropoids. The latter clade subsequently divides into monophasic New World primates (Platyrrhini) and Old World primates (Catarrhini). Platyrrhini display a tri-partite split between Strepsirrhini and Haplorrhini: the first involves Indriidae (Indri and relatives), the second involves Cercopithecidae (as the sister group of Cercopithecus, or is that of *Erythrocebus* + *Chlorocebus,* or is the most basal genus in the clade). The PI property is thus less often satisfied in the primate supertree, leading to a greater number of irresolutions in the *PhySIC* supertree.

Identifying and Labeling the Causes of Supertree Polytomies.—Because veto methods are used for evaluating the topological congruence of source trees, and for measuring their degree of leaf overlap, the *PhySIC* program outputs labels on each polytomous node. A label "C" (standing for *Contradiction*) indicates that the polytomy results from contradictions among the source trees on phylogenetic relationships of corresponding taxa: proposing a resolution for the polytomy would contradict at least one source tree; i.e., would not respect the PC property. A label "I" (standing for *Induction*) indicates a lack of cross-information in the source trees: any dichotomous resolution of the clade would be at least partially arbitrary and thus would not respect the PI property. Note that a given label applies only to the node to which it is assigned but not to other nodes in its subtrees. For instance, in the primate genera supertree (Figure 13), the platyrrhine trifurcation (Atelidae, Pitheciidae, Cebidae) with a C label indicates that there is topological contradiction among the source trees about the sister-group relationships of these three families. However, the C label does not put the monophyly of Atelidae, Pitheciidae, and Cebidae into question. Note also that a same polytomy can be characterized by both C and I labels. This means that the inability of the supertree to propose a dichotomous resolution is partly due to a lack of taxonomic overlap, and partly due to contradictions. For example, Figure 13 shows that the clade *Cercopithecus, Erythrocebus, Chlorocebus,* and *Miopithecus* is tagged by both C and I, reflecting two problems. On the one hand, source topologies disagree about the placement of *Erythrocebus:* this genus is either related to *Cercopithecus* (as suggested by IRBP exon 1) or to *Chlorocebus* (cf. the TSPY and chromosome Xq13.3 markers, and the ALU characters of Xing et al. [2005]). On the other hand, the input trees analyzed here do not provide the information required to know whether *Miopithecus* is the sister group of *Cercopithecus,* or is that of *Erythrocebus* + *Chlorocebus,* or is the most basal genus in the clade.

Impact of the Robustness of Source Trees on Veto Supertree Resolution.—The number of clades retained from the original source trees depends on the bootstrap threshold imposed to select them for supertree inference. Choosing a low threshold thus increases the number of retained source clades and hence lowers the number of polytomies due to a lack of cross-information among source trees but increases the number of polytomies due to conflicts among source trees. Increasing the threshold has the opposite effect. The primate supertree of Figure 13 was obtained with BP ≥ 70%. Lowering the threshold to BP ≥ 50% or BP ≥ 60%, *PhySIC* yields a completely multifurcating supertree, due to weakly supported clades that conflict among source trees. When the bootstrap stringency is increased from the BP ≥ 70% to BP ≥ 80% threshold, a similar level of resolution in the genus level phylogeny is obtained with the exception of two additional polytomies: the first involves Indriiidae (*Indri* + *Avahi* + *Propithecus* + *Paleopropithecus*) relative to other lemuriforms, and the second involves *Allenopithecus* relative to the *Cercopithecus* clade (white stars in Figure 13 refer to disappearing branches). Interestingly, increasing the threshold removes a topological conflict among *Lophocebus, Papio* and *Theropithecus:* with the PC property being satisfied, then the *PhySIC* supertree groups together the latter two genera. At the BP ≥ 90% threshold, 7 additional polytomies with respect to the BP ≥ 70% topology appear (Figure 13: black stars refer to node collapsing). This reflects the fact that less source nodes (i.e., the nodes of source trees) are available for supertree inference. The PI property is thus less often satisfied in the *PhySIC* supertree, leading to a greater number of irresolutions in the *PhySIC* supertree.

---

**Table 1.** Molecular markers used to infer the primate source trees. The best-fitting model and the number of primate genera per marker are indicated. (mtDNA: mitochondrial DNA)

<table>
<thead>
<tr>
<th>Marker</th>
<th>Model</th>
<th>Genera</th>
</tr>
</thead>
<tbody>
<tr>
<td>α-2B Adrenergic receptor</td>
<td>HKY + Γ</td>
<td>16</td>
</tr>
<tr>
<td>Albumin gene introns 3 and 4</td>
<td>HKY + I + Γ</td>
<td>20</td>
</tr>
<tr>
<td>ATPase 7A</td>
<td>HKY + Γ</td>
<td>11</td>
</tr>
<tr>
<td>Breast and ovarian cancer susceptibility</td>
<td>HKY + Γ</td>
<td>12</td>
</tr>
<tr>
<td>Cytochrome b [mtDNA]</td>
<td>GTR + Γ + I</td>
<td>68</td>
</tr>
<tr>
<td>α-1,2 lucosyltransferase</td>
<td>HKY + Γ</td>
<td>21</td>
</tr>
<tr>
<td>β globin exons 1 and 2</td>
<td>HKY + Γ</td>
<td>22</td>
</tr>
<tr>
<td>γ globin</td>
<td>HKY + Γ</td>
<td>26</td>
</tr>
<tr>
<td>Glucose-6-phosphate dehydrogenase introns 4 and 5</td>
<td>HKY + Γ</td>
<td>18</td>
</tr>
<tr>
<td>Glucose-6-phosphate dehydrogenase introns 7 and 8</td>
<td>HKY + Γ</td>
<td>12</td>
</tr>
<tr>
<td>Interphotoreceptor retinoid-binding protein exon 1</td>
<td>K80 + Γ</td>
<td>34</td>
</tr>
<tr>
<td>Interphotoreceptor retinoid-binding protein intron 1</td>
<td>K80 + Γ</td>
<td>24</td>
</tr>
<tr>
<td>Lysozyme gene</td>
<td>HKY + Γ</td>
<td>19</td>
</tr>
<tr>
<td>Membrane cofactor protein gene</td>
<td>HKY + Γ</td>
<td>13</td>
</tr>
<tr>
<td>α2-microglobulin precursor exons 1 and 2</td>
<td>HKY + Γ</td>
<td>18</td>
</tr>
<tr>
<td>NADH dehydrogenase subunit 5</td>
<td>GTR + Γ + I</td>
<td>24</td>
</tr>
<tr>
<td>[mtDNA] Phospholipase C β4 gene</td>
<td>GTR</td>
<td>13</td>
</tr>
<tr>
<td>Testis-specific protein gene</td>
<td>HKY + Γ</td>
<td>21</td>
</tr>
<tr>
<td>von Willebrand gene introns 11 and 12</td>
<td>HKY + Γ</td>
<td>37</td>
</tr>
<tr>
<td>9.3 kb chromosome Xq13.3 fragment</td>
<td>HKY + Γ</td>
<td>13</td>
</tr>
<tr>
<td>SINE [Roos et al., 2004]</td>
<td>—</td>
<td>20</td>
</tr>
<tr>
<td>ALU [Singer et al., 2003]</td>
<td>—</td>
<td>8</td>
</tr>
<tr>
<td>SINE [Xing et al., 2005]</td>
<td>—</td>
<td>15</td>
</tr>
</tbody>
</table>
Figure 13. Primate PhySIC supertree including 95% of all extant genera and containing no contradiction or arbitrary resolution with respect to the source trees, as defined by the PI and PC properties. The genus-level primate supertree has been reconstructed from 24 molecular source trees restricted to nodes supported by more than 70% bootstrap. When the bootstrap threshold is increased to 80%—respectively, 90%—the supertree topology changes: disappearing branches as well as one appearing clade (Papio + Theropithecus) are indicated by white—respectively, black—stars. Polytomies are labeled by tags pointing out the properties (PI, PC, or both) that would not be satisfied if the corresponding clade was more resolved. The taxonomic frame and clade names for Primates are given. Hatched rectangles represent Anthropoidea (Platyrrhini + Catarrhini). White and black rectangles respectively represent Haplorrhini (Tarsiiformes + Anthropoidea) and Strepsirrhini (Lorisiformes + Lemuriformes).
Overall, two reasons can lead the *PhysIC* method to propose a poorly resolved supertree. First, it is possible that the source trees contain too little cross-information for the method to decide how the taxa of the respective source trees branch relatively to each other. In this case, all methods, including voting methods, will produce unresolved supertrees. Obtaining more resolved supertrees can then only be achieved by adding new source trees containing new clades on the key taxa. The second reason why the *PhysIC* supertree can lack resolution is the presence of topological conflicts among source trees. Like other veto methods, *PhysIC* is very sensitive to incongruences in the source trees. Thus, to obtain a well-resolved tree, a preliminary process whereby unreliable clades are collapsed in the source trees is usually necessary before applying the method. This collapsing can be done on the basis of the support values provided on the clades by most phylogenetic inference methods (e.g., bootstrap values, Bayesian posterior probabilities, Bremer support). We showed that a well-resolved supertree of Primates can be obtained with such an approach from a nontrivial number of gene trees. Note that on some data sets, contradicting clades showing high support values can occur, e.g., due to lateral gene transfers. In such cases, veto methods will still produce unresolved supertrees (as long as they are not allowed to exclude rogue taxa). This can be seen as a drawback or as a way to pinpoint such events. In such cases, outlier source trees can be identified (Shimodaira and Hasegawa, 1999; Lerat et al., 2003) and then curated or removed from the collection of source trees, leading to a more resolved supertree.

**CONCLUSION**

Veto supertree methods are of interest for combining source topologies containing reliable clades. Their study also brings insight for the characterization of what we expect from voting methods. Indeed, when source trees are not conflicting, there is no fundamental difference between the two approaches. In such cases, veto and voting approaches should lead to reasonable supertrees. What reasonable means can be characterized by several formal properties. In the present work, we showed pitfalls of some previously proposed supertree properties, and also proposed new properties. In the general case of conflicting source trees, we believe there is still room for improvement, e.g., detecting arbitrary clades of a supertree even when it partially conflicts with some source trees, as usually happens in the voting context. With the new theoretical material at hand we believe that this is a reasonable goal.

**ACKNOWLEDGEMENTS**

We thank J. Cotton, R. Page, O. Bininda-Emonds, and an anonymous reviewer for many invaluable remarks on a first version of this manuscript. This work has been supported by the “ACI Informatique-Mathématique-Physique en Biologie Moléculaire [ACI IMP-Bio],” by the “Action incitative BIOSTIC-LR,” by IFR119 “Biodiversité Continentale Méditerranéenne et Tropicale” (Montpellier), and by the Research Networks Program in BIOINFORMATICS of the High Council for Scientific and Technological Cooperation between France and Israel. This publication is the contribution no. 2007-053 of the Institut des Sciences de l’Évolution de Montpellier (UMR 5554-CNRS). We thank V. Lefort for creating the web site of *PhysIC*.

**REFERENCES**


First submitted 21 August 2006; revisions returned 23 November 2006; final acceptance 8 June 2007

Associate Editor: Olaf Bininda-Emonds

APPENDIX 1

Proof of Proposition 1

The proof is based on the fact that a set R identifies a tree T if and only if r(T) = cl(R) (Grunewald et al., 2006, lemma 2.1).

⇒ By definition, R(T, T) only contains triplets on three-taxon sets for which there is a resolution in T. Moreover, PC ensures that any such triplet cannot be resolved in R(T, T) differently than in T. It follows that R(T, T) ⊆ r(T). (R(T, T) is therefore compatible and cl(R(T, T)) ≤ cl(r(T)) = r(T)). Meanwhile, having proved that R(T, T) is compatible, it is clear from PI that cl(r(T, T)) ≥ r(T).

⇐ Having R(T, T) identifying T ensures that R(T, T) is compatible. On one hand, cl(R(T, T)) = r(T) implies that for all t ∈ r(T), t ∈ cl(R(T, T)); i.e., R(T, T) ∩ t and therefore PI holds. On the other hand, given t ∈ R(T, T), then, by definition of the closure, I ∈ cl(R(T, T)) = r(T) so that both I and t are in r(T), which is not possible. This proves that R(T, T) ⊄ t for any t ∈ r(T); i.e., PC holds.
Proof of Proposition 3

Let $ABIC \in r(T)$ be a triplet of the output supertree $T$ and consider the recursive step where the tree returned in line 3 hosts $A$, $B$ in the same subtree $T_i$, whereas $C$ is in another subtree, say $T_j$. This means that $A$, $B$ are vertices in a connected component $C_i$ of the current Aho graph $G$, whereas $C$ is in another component $C_j$ of $CC(G)$. If $R$ contained $AC \mid B$ or $BC \mid A$, then $C_i$ and $C_j$ would not have been distinct connected components (in such cases, graph $G$ would have contained the edge $(A, C)$ or $(B, C)$). Thus, for any triplet $t$ of $r(T)$, we know that no triplet $I$ is in $R$; i.e., $T$ does not directly contradict $R$. This ensures that $Build_{PC}$ returns a tree satisfying PC (lemma 1).

Proof of Theorem 1

Correctness of the Algorithm.—The triplets present in the tree $T_{PC}$ returned by $PhySIC_{PC}$ depend on the internal branches of this tree. Any internal branch of $T_{PC}$ is created at line 10 during some recursive step, thus linking a subtree $T_i$ to the root of the tree returned by this step. Thanks to Lemma 1, we just have to prove that every branch created at this line does not generate a triplet that directly contradicts $R$.

At this line, either (i) $CC_{PC}$ corresponds exactly to the connected components of $G$ (we have $|CC(G)| > 1$) and then, from Proposition 3, the triplets created are not in direct contradiction with $R$; or (ii) $CC_{PC}$ corresponds to a partition of the vertices of $G$ based on $R$. Two cases are then possible. In the first case (line 4), $G$ is connected and the current call returns a multifurcation that generates no triplet and hence does not invalidate PC. In the second case, the $Repeat$ loop ensures that for each set of three taxa $A, B, C \in V(G)$, $CC_{PC}$ does not contain two elements $C_i$ and $C_j$ with $A, B \in C_i$, and $C \in C_j$ such that either $AC \mid B$ or $BC \mid A$ belongs to $R$ or $R \cup R_{PC}$. Indeed, $AC \mid B$ (and $BC \mid A$) cannot be in $R_{PC}$ since $C_i$ would have been removed from $CC_{PC}$ (line 9). Moreover, if $AC \mid B$ (respectively $BC \mid A$) was in $R'$, then $A$ and $C$ (respectively $B$ and $C$) would have been in the same component of $CC_{PC}$ contradicting $C_i \neq C_j$. This proves that the tree $T_{PC}$ built in line 10, and whose subtrees bijectively correspond to the elements of $CC_{PC}$, is such that no triplet of $r(T_{PC})$ directly contradicts $R$.

Time Complexity of the Algorithm.—As for $PhySIC_{PC}$, the most time-consuming operations done by $PhySIC_{PC}$ are the construction of the Aho graph $G_i$, the computation of its connected components in the $Check_{PC}$ subroutine. The $G_i$ graphs that may be used in $Check_{PC}$ can be precomputed in the $PhySIC_{PC}$ part of the pseudo-code (i.e., before calling $Check_{PC}$), knowing $r(T)$ and the current tree $T_{PC}$ to be examined in $Check_{PC}$. This preprocessor clearly requires $O(n^2)$ time, since there are $O(n)$ such graphs (one for each clade of $T$), each of which is obtained by examining the $O(n^3)$ graph of $R(T_{PC})$.

The preprocesses done for $G_{ij}$ graphs thus require $O(n^3)$ time. As this has to be done at each pass of the $Repeat$ loop, the only information used by $Check_{PC}$ on graph $G_{ij}$ is the number of their connected components. The total number of edges present in the $G_i$ graphs is in $O(n^2)$: precomputation of the number of connected components for this set of graphs is thus globally $O(n^2)$ time. This is done at most $O(n)$ times (each pass results in the collapsing of one of the $O(n)$ clades of $T$), this part of the computation is globally (on the whole for $PhySIC_{PC}$) in $O(n^2)$ time. Determination of the number of connected components of each $G_i$ is done only once just before the $Repeat$ loop. For each of these $O(n)$ graphs, this requires examining $O(n^2)$ triplets. Thus, this preprocessor also costs $O(n^3)$ time.

The preprocesses done for $G_i$ and $G_{ij}$ graphs thus require $O(n^3)$ time and reduce the running time of $Check_{PC}$. The modification of $Check_{PC}$, consisting of returning to $PhySIC_{PC}$ as soon as an edge is collapsed, also simplifies the algorithm (e.g., the $Repeat$ loop is no longer required).

Thanks to the preprocessing, the only time-consuming operation in $Check_{PC}$, for the current tree $T_{PC}$, is the examination of the $O(n^2)$ pairs of sibling clades $C_i$ and $C_j$ of this tree. Operations performed for each of this pair of clades is in $O(1)$ (the number of connected components of useful graphs $G_i$, $G_j$, and $G_{ij}$ have been preprocessed). Because a new tree $T_{PC}$ can only be obtained by collapsing one of the $O(n)$ edges (line 14) of $T_{PC}$, this can at most occur $O(n)$ times. Therefore, all executions of $Check_{PC}$ issued by a run of $PhySIC_{PC}$ are in $O(n^3)$ time. Thus, the whole complexity of the $PhySIC_{PC}$ algorithm is no more than the cost of the preprocessing; i.e., $O(n^3)$ time.